Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2007

D. Harkins, Ed.
The Industrial Lounge

June 29, 2007

# SIV Authenticated Encryption using AES
# draft-dharkins-siv-aes-00

## Status of this Memo



## Copyright Notice



## Abstract

This memo describes SIV, a block cipher mode of operation. SIV takes a key, a plaintext, and a vector of data which will be authenticated but not encrypted. It produces a ciphertext having the same length as the plaintext and a synthetic initialization vector. Depending on how it is used, SIV achieves either the goal of deterministic authenticated-encryption or the goal of nonce-based, misuse-resistant authenticated-encryption.

Table of Contents

# 1. Introduction

## 1.1. Background

Various attacks have been described (e.g. [BADESP]) when data is merely privacy-protected and not additionally authenticated or integrity protected. Therefore combined modes of encryption and authentication have been developed ([GCM], [JUTLA], [CCM], [OCB], [AEAD]). These provide conventional, probabilistic authenticated-encryption when used with a nonce ("a number used once") and typically accept additional inputs that are authenticated but not encrypted.

A deterministic, nonce-less, form of authenticated-encryption has been used to protect the transportation of cryptographic keys (e.g. [X9F1], [RFC3217], [RFC3394]). This is generally referred to as "Key Wrapping".

This memo descirbes a new block cipher mode, SIV, that provides both probabilistic, nonce-based authenticated encryption as well as deterministic, nonce-less key wrapping. It contains a PRF construction called S2V. Both S2V and SIV were specified by Phillip Rogaway and Thomas Shrimpton [DAE]. The underlying block cipher used herein for both S2V and SIV is AES. S2V uses AES-CMAC and will be referred to as AES-CMAC-AES and SIV uses AES-CTR and will be referred to as SIV-CTR-AES.

## 1.2. Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 1.3. Motivation

### 1.3.1. Key Wrapping

A key distribution protocol must protect keys it is distibuting. This has not always been done right. For example RADIUS [RFC2548] uses MPPE to encrypt a key prior to transmission from server to client. It provides no integrity checking of the encrypted key. [RADKEY] specifies the use of [RFC3394] to wrap a key in a RADIUS [RFC2865] request but because of the inability to pass additional authenticated data an HMAC [RFC2104] is necessary to provide authentication of the entire request.

SIV can be used as a drop-in replacement for any specification that uses [RFC3394] or [RFC3217], including the aforementioned use. It is





a more general purpose solution as it allows for additional authenticated data to be specified.

### 1.3.2. Resistance to Nonce Misuse/Reuse

The probabilistic, nonce-based authenticated encryption schemes described above are susceptible to reuse and/or misue of the nonce. Depending on the specific scheme there are subtle and critical requirements placed on the nonce or counter. [GCM] states that it provides "excellent security" if its initialization vector (IV) is guaranteed to be distinct but provides "no security" otherwise. Confidentiality guarantees are voided if a counter in [CCM] is reused. In many cases guaranteeing no reuse of a nonce/counter/IV is not a problem but in others it will be. For example, if one's environment is (knowingly or unknowingly) a virtual machine it may be possible to roll back a virtual state machine and cause nonce reuse thereby gutting the security of the authenticated encryption scheme (see [VIRT]).

Also, if the nonce is random a requirement that it be non-repeating will dramatically limit the amount of data that can be safely protected with a single key.

SIV provides a level of resistance to nonce reuse and misuse. If the nonce is never reused then the usual notion of nonce-based security of an authenticated encryption mode is achieved. If, however, the nonce is reused authenticity is retained and confidentiality is only compromised to the extent that an attacker can determine that the same plaintext (and same additional authenticated data) was protected

with the same nonce and key. See Security Considerations (Section 7).

### 1.3.3. Key Derivation

A PRF is frequently used as a key derivation function (e.g. [WLAN]) by passing it a key and a single string. Typically this single string is the concatenation of a series of smaller strings-- for example, a label and some context to bind into the derived string.

These strings are logically a vector of strings but are mapped to a single string because of the way PRFs are typically defined-- two inputs: a key and data. Such a crude mapping is inefficient because additional data must be included-- the length of inputs must be encoded separately-- and, depending on the PRF, memory allocation and copying is needed. Also, if only one or two of the inputs changed when deriving a new key it may still be necessary to process all of the other constants that preceded it every time the PRF is invoked.





When a PRF is used in this manner its input is a vector of strings and not a single string and the PRF should handle the data as such. The S2V ("string to vector") PRF construction accepts a vector of inputs and provides a more natural mapping of input that does not require additional lengths encodings and obviates the memory and processing overhead to marshall inputs and their encoded lengths into a single string. Constant inputs to the PRF need only be computed once.

### 1.3.4. Robustness versus Performance

SIV can not perform at the same high troughput rates that other authenticated encryption schemes can (e.g. [GCM] or [OCB]) but for situations where performance is not a limiting factor-- e.g. control plane applications-- it can provide a robust alternative.

## 2. Specification of SIV-AES

### 2.1. Notation

SIV and S2V use the following notation:

len(A)
: returns the number of bits in A.

X10*
: indicates padding of string X, len(X) < 128, out to 128 bits by the concatenation of a single bit of 1 followed by as many 0 bits as are necessary.

leftmost(A,n)
: the n most significant bits of A.

rightmost(A,n)
: the n least significatn bits of A.

A || B
: means concatenation of string A with string B.

A xor B
: is the exclusive OR operation on two equal length strings, A and B.

A xorend B
: where len(A) >= len(B), means xoring a string B onto the end of string A-- i.e. leftmost(A, len(A)-len(B)) || (rightmost(A, len(B)) xor B)





dbl(S)
: is the multiplication of S and 0...010 in a finite field represented using the primitive polynomial x^128 + x^7 + x^2 + x + 1. See Doubling (Section 2.3)

<zero>
: indicates a string represented by 128 zero bits.

<one>
: indicates a string represented by 127 zero bits concatenated with a single one bit.

E(K,X)
: indicates AES encryption using key K of 128-bit string X

## 2.2. Overview

SIV-AES uses AES in CTR mode, called SIV-CTR-AES, and a pseudo random function (PRF) based on AES-CMAC called S2V-CMAC-AES. SIV-AES takes either a 256, 384, or 512 bit key which is broken up into two equal-sized keys, one for S2V-CMAC-AES and the other for SIV-CTR-AES.

## 2.3. Doubling

The doubling operation on an input string is performed using a left-shift of the input followed by a conditional xor operation on the result with the constant:

00000000 00000000 00000000 00000087

The condition under which the xor operation is performed is when the bit being shifted off is one.

Note that this is the same operation used to generate sub-keys for AES-CMAC

## 2.4. S2V-CMAC-AES

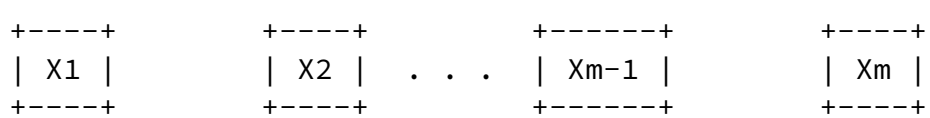
The S2V-AES-CMAC operation consists of the doubling and xoring of the outputs of AES-CMAC operations over individual strings in the input vector. The operation is bootstrapped by performing AES-CMAC on a 128-bit string of zeros. If the length of the final string in the vector is greater than or equal to 128 bits the doubled and xored output is xored onto the end of the final input string. That result is input to a final AES-CMAC operation to produce the output Z. If the length of the final string is less than 128 bits the doubled and xored output is doubled once more and it is xored with the final string padded with 10* up to 128 bits. That result is input to a

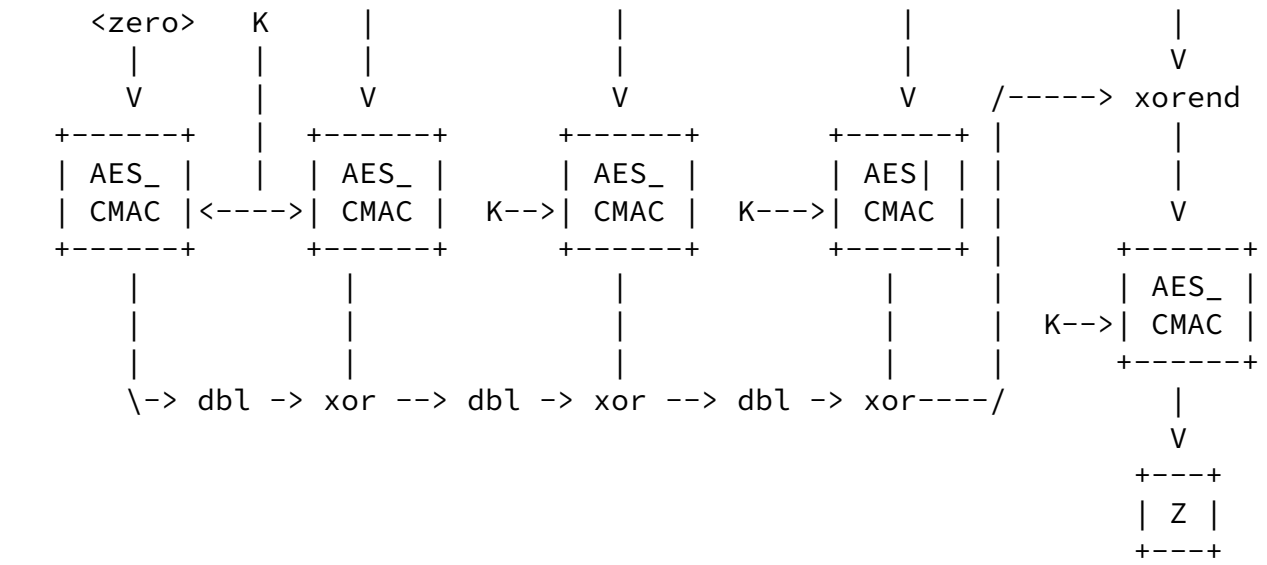




final AES-CMAC operation to produce the output Z.

S2V-AES-CMAC with key k on a vector of m inputs X1, X2, ..., Xm-1, Xm, and len(Xm) >= 128:






where 'dbl' is the double operation

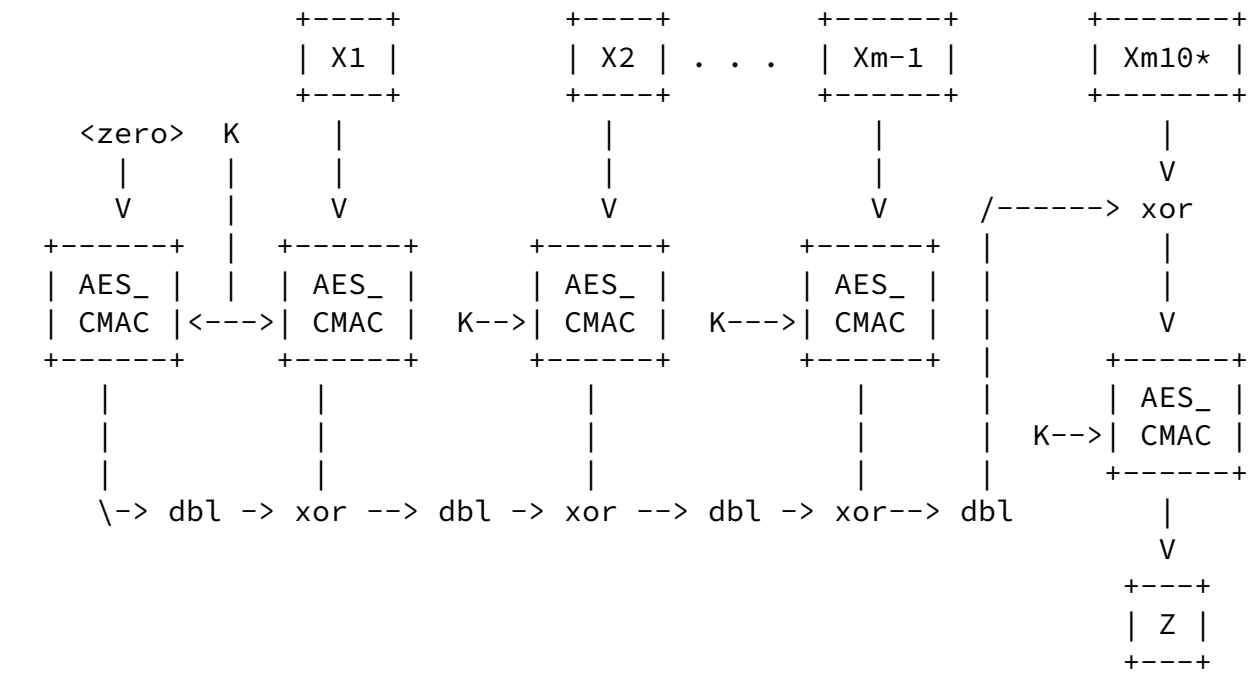
Figure 2





S2V-AES-CMAC with key k on a vector of m inputs X1, X2, ..., Xm-1,

Xm, and len(Xm) < 128:




where 'dbl' is the double operation

Figure 3

Algorithmically S2V-AES-CMAC can be described as:

```
S2V-AES-CMAC(K, X1, ..., Xm) {
  if m = 0 then
    return AES-CMAC(K, <one>)
  fi
  S <-- AES-CMAC(K, <zero>)
  for i = 1 to m-1 do
    S <-- dbl(S) xor AES-CMAC(K, Xi)
  done
  if len(Xm) >= 128 then
    T <-- Xm xorend S
  else
    T <-- dbl(S) xor Xm10*
  fi
  return Z <-- AES-CMAC(T)
}
```

## 2.5. SIV-CTR-AES

SIV-AES-CTR is a counter mode of AES. It takes as input a plaintext, P, that is less than [(2^32 - 1) * 128] bits, a key K of length 256, 384 or 512 bits, and a counter CTR that is 128 bits in length, and outputs Z which represents a concatenation of the synthetic initialization vector SIV, and the ciphertext, C, which is the same length as the plaintext. The size limitation on the plaintext is a tradeoff made for efficient incrementing of the counter.

When len(K) is 256 bits then the underlying AES cipher uses a 128 bit key; when len(K) is 384 bits then the underlying AES cipher uses a 192 bit key; and, when len(K) is 512 bits then the underlying AES cipher uses a 256 bit key.

The ciphertext is produced by xoring the plaintext with the first len(P) bits of the following string:

E(K, CTR) || E(K, CTR+1) || E(K, CTR+2) || ...

The increment function is handled by treating the initial counter as 96 bits of constant salt followed by a 32 bit non-negative integer which is incremented modulo 2^32. More formally,

SALT=leftmost(CTR,96)

n=rightmost(CTR,32)

CTR+i = SALT || (n + i mod 2^32).

## 2.6. SIV-AES Encrypt

SIV-AES-encrypt takes as input a key K of length 256, 384 or 512 bits, plaintext of length less than [(2^32 - 1) * 128] bits, and additional data which is authenticated but not encrypted. It produces output, Z, which is the concatenation a 128 bit synthetic IV and ciphertext whose length is equal to the length of the plaintext.

The key is split into two, K1 = leftmost(K, len(K)/2) and K2 = rightmost(K, len(K)/2). K1 is used for S2V-AES-CMAC and K2 is used for AES-CTR.

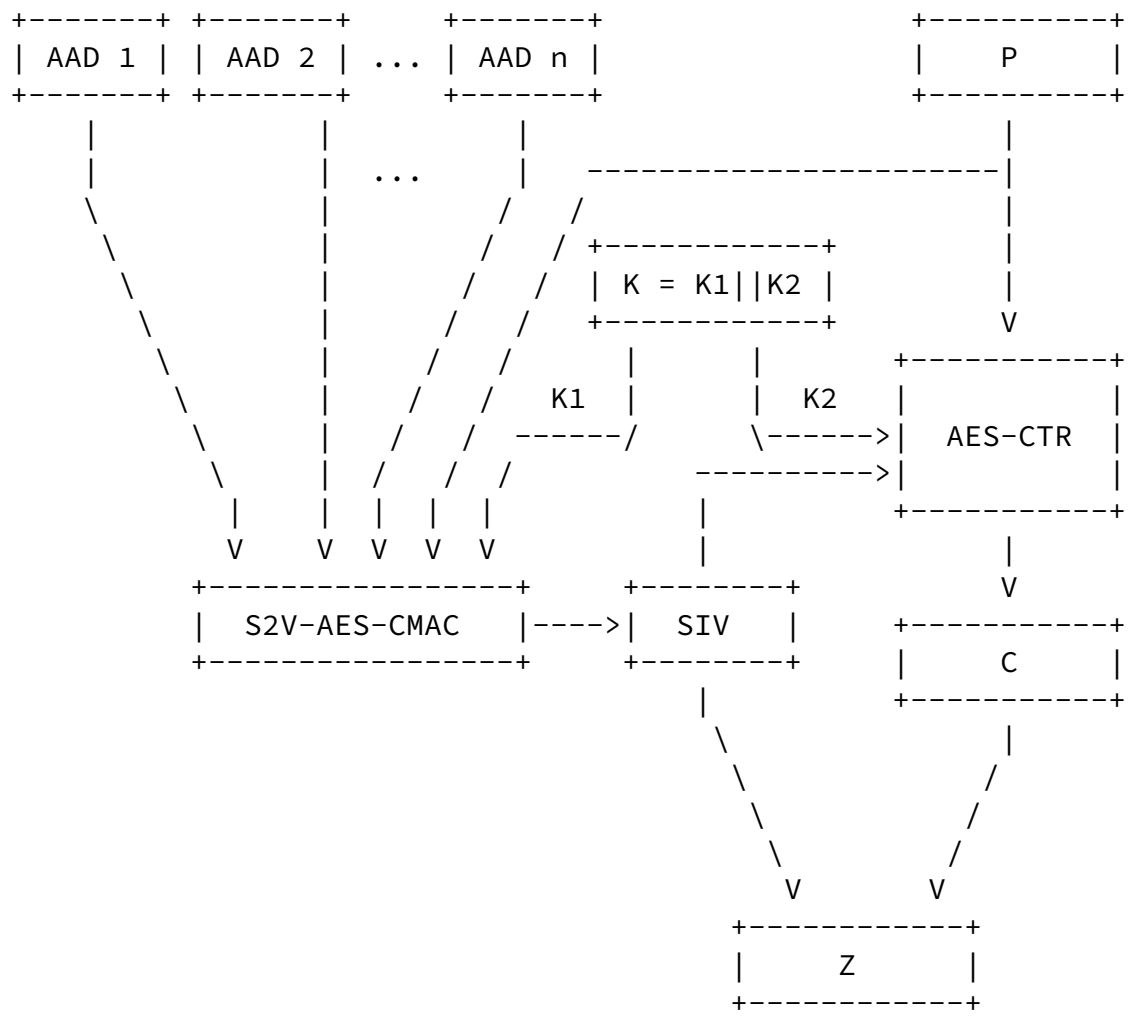
In the encryption mode the additional authenticated data and plaintext represent the vector of inputs to S2V-AES-CMAC, with the plaintext being the last string in the vector. The output of S2V-AES-CMAC is a synthetic IV which represents the initial counter used

on the plaintext with AES-CTR mode.





The encryption construction of SIV is as follows:




where the plaintext is P, the associated data is AAD1 through AADn, SIV is the synthetic IV, the ciphertext is C, and Z is the output.

Figure 7

Algorithmically SIV-AES Encrypt can be described as:

```
SIV-AES-ENCRYPT(K, P, AAD1, ..., AADn) {
  K1 <-- leftmost(K, len(K)/2)
  K2 <-- rightmost(K, len(K)/2)
  V <-- S2V-AES-CMAC(K1, AAD1, ..., AADn, P)
  m = (len(P) + 127)/128

  for i = 0 to m-1 do
    X <-- AES(K2, V+i)
    Ci <-- Pi xor X
  done
  C <-- C1, ... Cm

  return V, C
}
```

## 2.7. SIV-AES Decrypt

SIV-AES-decrypt takes as input a key K of length 256, 384 or 512 bits, Z which represents a synthetic initialization vector SIV concatentated with a ciphertext C, and additional data which is authenticated but not encrypted. It produces either the original plaintext or the special symbol FAIL.

The key is split as specified in Section 2.6

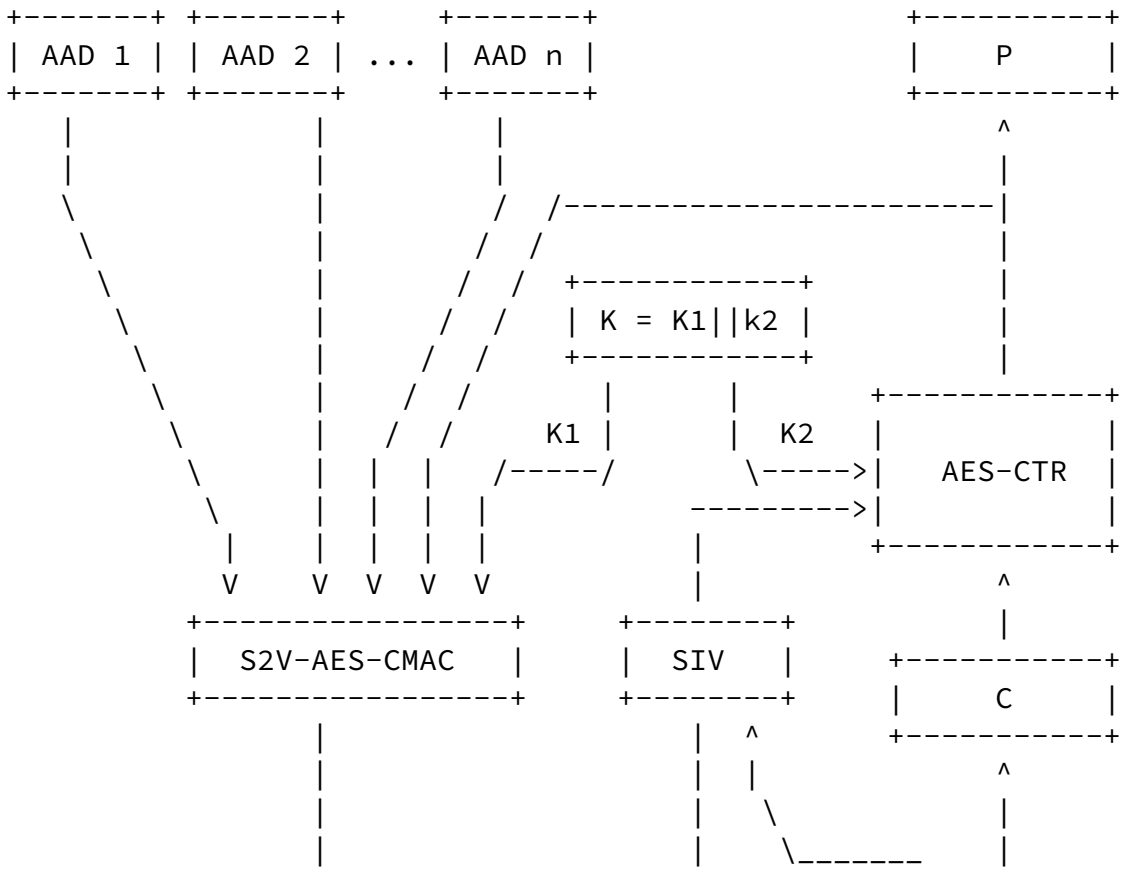
The synthetic IV acts as the initial counter to AES-CTR mode to decrypt the ciphertext. The additional authenticated data and the output of AES-CTR mode is used to represent the vector of inputs to S2V-AES-CMAC, with the AES-CTR mode output being the last string in the vector. The output of S2V-AES-CMAC is then compared against the

synthetic IV that accompanied the original ciphertext. If they match the output from AES-CTR mode is returned as the decrypted and authenticated plaintext otherwise the special symbol FAIL is returned.





The decryption construction of SIV is as follows:



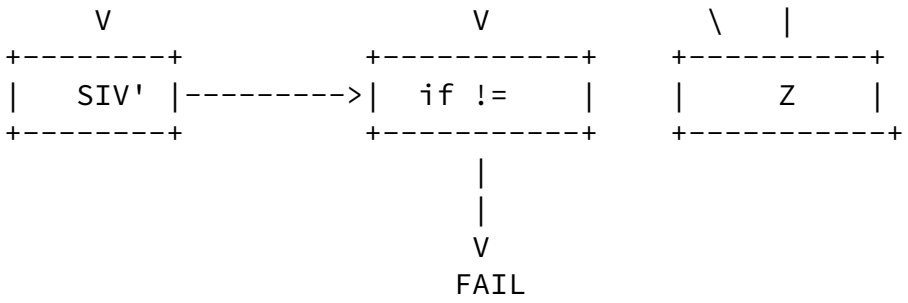



Figure 9





Algorithmically SIV-AES Decrypt can be described as:

```
SIV-AES-DECRYPT(K, C, V, AAD1, ..., AADn) {
  K1 <-- leftmost(K, len(K)/2)
  K2 <-- rightmost(K, len(K)/2)

  m = (len(C) + 127)/128
  for i = 0 to m-1 do
    X <-- AES(K2, V+i)
    Pi <-- Ci xor X
  done
  P <-- P1, ... Pm
  T <-- S2V-AES-CMAC(K1, AAD1, ..., AADn, P)

  if T = V then
    return P
  else
    return FAIL
```

```
  fi
}
```

## 3. Nonce-based Authenticated Encryption with SIV-AES

SIV-AES performs nonce-based authenticated encryption when a component of the additional authenticated data is a nonce. For purposes of interoperability the final component-- i.e. the string immediately preceding the plaintext in the vector input to S2V-AES-CMAC-- is used for the nonce. Other additional authenticated data are optional.

If the nonce is random it SHOULD be at least 128 bits in length and be harvested from a pool having at least 128 bits of entropy. A non-random source MAY also be used, for instance a time stamp. The definition of a nonce precludes reuse but SIV-AES is resistant to nonce reuse. See Section 1.3.2 for a discussion on the security implications of nonce reuse.

It MAY be necessary to transport this nonce with the output generated by S2V-AES-CMAC.

## 4. Deterministic Authenticated Encryption with SIV-AES

When the plaintext to encrypt and authenticate contains a nonce itself SIV-AES can be used in a deterministic mode to perform "key wrapping". Because S2V-AES-CMAC allows for additional authenticated data and imposes no unnatural size restrictions on the data it is





protecting (the only requirement being it is less than [(2^32 - 1) * 128] bits) it is a more useful and general purpose solution than [RFC3394]. Protocols which use SIV-AES for deterministic authenticated encryption (i.e. for more than just wrapping of keys) MAY define additional authenticated data inputs to SIV-AES. No nonce is necessary in this mode.

## 5. Optimizations

It is possible to optimize an implementation of S2V-AES-CMAC when it

is being used as a key derivation function (KDF), for example in [WLAN]. This is because the S2V construct operates on a vector of distinct strings and typically the data passed to a KDF contains constant strings. Depending on the location of the varient component of the input the AES-CMAC'd output of intermediate and invarient components can be computed once and xor'd with the running sum or an intermediate value of the doubled and xor'd output up to the varient component can be computed once and cached.

## 6. IANA Considerations

[AEAD] defines a uniform interface to cipher modes which provide nonce-based authenticated encryption with additional authentication data (AEAD). It does this via a registry of AEAD algorithms.

The Internet Assigned Numbers Authority (IANA) will assign three entries from the AEAD Registry for SIV-AES-256, SIV-AES-384, and SIV-AES-512 based upon the following AEAD algorithm definitions. The security analysis for each of these algorithms is in [DAE].

### 6.1. AEAD_SIV_AES_256

The SIV-AES-256 AEAD algorithm works as specified in Section 2.6 and Section 2.7. The input and output lengths for SIV-AES-256 as defined by [AEAD] are:

K_LEN is 32 octets.

P_MAX is 2^35 octets.

A_MAX is unlimited.

N_MIN is 1 octet.





N_MAX is unlimited.

C_MAX is 2^35 + 16 octets.

The security implications of nonce re-use and/or mis-use are described in Section 1.3.2.

## 6.2. AEAD_SIV_AES_384

The SIV-AES-384 AEAD algorithm works as specified in Section 2.6 and Section 2.7. The input and output lengths for SIV-AES-384 as defined by [AEAD] are:

K_LEN is 48 octets.

P_MAX is 2^35 octets.

A_MAX is unlimited.

N_MIN is 1 octet.

N_MAX is unlimited.

C_MAX is 2^35 + 16 octets.

The security implications of nonce re-use and/or mis-use are described in Section 1.3.2.

## 6.3. AEAD_SIV_AES_512

The SIV-AES-512 AEAD algorithm works as specified in Section 2.6 and Section 2.7. The input and output lengths for SIV-AES-512 as defined by [AEAD] are:

K_LEN is 64 octets.

P_MAX is 2^35 octets.

A_MAX is unlimited.

N_MIN is 1 octet.

N_MAX is unlimited.

C_MAX is 2^35 + 16 octets.

The security implications of nonce re-use and/or mis-use are described in Section 1.3.2.

## 7. Security Considerations

SIV-AES provides privacy in the sense that the output of SIV-AES Encrypt is indistinguishable from a random string of bits. It provides authenticity in the sense that an an attacker is unable to construct a string of bits that will return other than FAIL when input to SIV-AES Decrypt. A proof of the security of SIV with an "all in one" notion of security for an authenticated encryption scheme is provided in [DAE].

SIV-AES in the deterministic authenticated encryption mode provides this sense of privacy and authenticity. In the deterministic mode a nonce component is added to the plaintext. Even when this nonce is made available to an attacker the output of SIV-AES Encrypt is indistinguishable from random bits. Similarly, even when this nonce is made available to an attacker she is unable to construct a string of bits that when input to SIV-AES Decrypt will return a plaintext encoded with the nonce-- i.e. it will only return FAIL.

When the nonce used in the nonce-based authenticated encryption mode of SIV-AES is is treated with the care afforded a nonce or counter in other probabilistic authenticated encryption schemes-- i.e. guarantee that it will never be used with the same key for two distinct invocations-- then SIV-AES achieves the level of security described above. If, however, the initialization vector is reused SIV-AES continues to provide the level of authenticity described above but with a slightly reduced amount of privacy (see Section 1.3.2).

## 8. Acknowledgments

Thanks to Phil Rogaway for patiently answering numerous questions on SIV and S2V and for useful critques of initial versions of this paper. Thanks also to David McGrew for numerous helpful comments and suggestions for improving this paper. Thanks to Jouni Malinen for producing another independent implementation of S2V and thereby confirming the correctness of the test vectors.

## 9. References

### 9.1. Normative References

[DAE] Rogaway, P. and T. Shrimpton, "Deterministic Authenticated Encryption, A Provable-Security Treatment of the Key-Wrap Problem", September 2006.

Requirement Levels", BCP 14, RFC 2119, March 1997.

## 9.2. Informative References

[AEAD] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", Internet-Draft: draft-mcgrew-auth-enc-02.txt (a work in progress), February 2007.

[BADESP] Bellovin, S., "Problem Areas for the IP Security Protocols", July 1996.

[CCM] Whiting, D., Housley, R., and N. Ferguson, "Counter With CBC-MAC (CCM)", June 2002.

[GCM] McGrew, D. and J. Viega, "The Galois/Counter Mode of Operation (GCM)".

[JUTLA] Jutla, C., "Encryption Modes With Almost Free Message Integrity", Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptography.

[OCB] Korvetz, T. and P. Rogaway, "The OCB Authenticated Encryption Algorithm", Internet-Draft: draft-krovetz-ocb-00.txt (a work in progress).

[RADKEY] Zorn, G., "RADIUS Attributes for the Delivery of Keying Material", Internet-Draft: draft-zorn-radius-keywrap-13.txt (a work in progress).

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", February 1997, <http://www.ietf.org/rfc/rfc2104.txt>.

[RFC2548] Zorn, G., "Microsoft Vendor-specific RADIUS Attributes", March 1999, <http://www.ietf.org/rfc/rfc2548.txt>.

[RFC2865] Rigney, C., Williams, S., Rubens, A., and W. Simpson,

"Remote Authentication Dial In User Service", June 2000, <http://www.ietf.org/rfc/rfc2865.txt>.

[RFC3217] Housley, R., "Triple-DES and RC2 Key Wrapping", December 2001, <http://www.ietf.org/rfc/rfc3217.txt>.

[RFC3394] Housley, R., "AES Key Wrap", February 2005, <http://www.ietf.org/rfc/rfc3394.txt>.





[VIRT] Garfinkel, T. and M. Rosenblum, "When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments".

[WLAN] "Draft Standard for IEEE802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification", 2007.

[X9F1] Dworking, M., "Wrapping of Keys and Associated Data", Request for review of key wrap algorithms. Cryptology ePrint report 2004/340, 2004. Contents are excerpts from a draft standard of the Accredited Standards Committee, X9, entitled ANS X9.102.

## Appendix A. Test Vectors

### A.1. Deterministic Authenticated Encryption Example

```
Input:
-----
Key:
        fffefdfc fbfaf9f8 f7f6f5f4 f3f2f1f0
        f0f1f2f3 f4f5f6f7 f8f9fafb fcfdfeff

AAD:
        10111213 14151617 18191a1b 1c1d1e1f
        20212223 24252627

Plaintext:
        11223344 55667788 99aabbcc ddee

S2V-AES-CMAC
```

```
   ------------
   CMAC(zero):
           0535e2dc b1e95ad2 3b168837 c2a2430b

   double():
           0a6bc5b9 63d2b5a4 762d106f 85448616

   CMAC(aad):
           f1f922b7 f5193ce6 4ff80cb4 7d93f23b

   xor:
           fb92e70e 96cb8942 39d51cdb f8d7742d

   double():
           f725ce1d 2d971284 73aa39b7 f1aee8dd
```





```
   pad:
           11223344 55667788 99aabbcc ddee8000

   xor:
           e607fd59 78f1650c ea00827b 2c4068dd

   CMAC(final):
           38c07e2c 86fc416d 18cfa186 7832f0fa

   SIV-AES-CTR
   -----------
   CTR:
           38c07e2c 86fc416d 18cfa186 7832f0fa

   E(K,CTR):
           60caaec0 312b627d 934b1293 5840ce08

   ciphertext:
   ----------
           71e89d84 644d15f5 0ae1a95f 85ae
```

## A.2. Probabilistic Authenticated Encryption Example

```
   Input:
   -----
   Key:
```

```
        7f7e7d7c 7b7a7978 77767574 73727170
        40414243 44454647 48494a4b 4c4d4e4f

AAD1:
        00112233 44556677 8899aabb ccddeeff
        deaddada deaddada ffeeddcc bbaa9988
        77665544 33221100

AAD2:
        10203040 50607080 90a0

IV:
        09f91102 9d74e35b d84156c5 635688c0

Plaintext:
        74686973 20697320 74686520 706c6169
        6e746578 7420746f 20656e63 72797074
        20757369 6e672053 49562d41 4553

S2V-AES-CMAC
------------
CMAC(zero):
```





```
        ba64ea67 710db6de ebdb99bd 08cc8c45

double():
        74c9d4ce e21b6dbd d7b7337a 1199180d

CMAC(aad1)
        3c9b689a b41102e4 80954714 1dd0d15a

xor:
        4852bc54 560a6f59 5722746e 0c49c957

double():
        90a578a8 ac14deb2 ae44e8dc 189392ae

CMAC(aad2)
        d98c9b0b e42cb2d7 aa98478e d11eda1b

xor:
        4929e3a3 48386c65 04dcaf52 c98d48b5
```

```
   double():
           9253c746 9070d8ca 09b95ea5 931a916a

   CMAC(IV)
           128c62a1 ce3747a8 372c1c05 a538b96d

   xor:
           80dfa5e7 5e479f62 3e9542a0 36222807

   xorend:
           74686973 20697320 74686520 706c6169
           6e746578 7420746f 20656e63 7279f0ab
           85922d2e f1051ec6 0bf61b63 6d54

   CMAC(final)
           efa831fb c6eb3ba8 84b81f30 ed59225e

   SIV-AES-CTR
   -----------
   CTR:
           efa831fb c6eb3ba8 84b81f30 ed59225e

   E(K,CTR):
           5fc2a9a6 b95e341c 6497f5e5 026eb7fa

   CTR+1:
           efa831fb c6eb3ba8 84b81f30 ed59225f
```





```
   E(K,CTR+1):
           6404c208 74585e4f 15b3d6a0 4f7e70f0

   CTR+2
           efa831fb c6eb3ba8 84b81f30 ed592260

   E(K,CTR+2):
           130a7acb f337caaf a06c1eac b2d60acc

   ciphertext:
   -----------
           2baac0d5 9937473c 10ff90c5 7202d693
```

```
            0a70a770 00782a20 35d6b8c3 3d070084
            337f09a2 9d50eafc e93a33ed f785
```

## Author's Address


Dan Harkins (editor)
The Industrial Lounge

Email: dharkins@lounge.org






## Full Copyright Statement

## Intellectual Property



## Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).